Design of Routing Protocols and Overlay Topologies for Topic-based Publish/Subscribe on Small-World Networks

Chen Chen
IBM Research - Haifa

Yoav Tock
IBM Research - Haifa

ABSTRACT

It is primarily important and challenging to develop a distributed publish/subscribe (pub/sub) system for large-scale workloads. On the one hand, structured overlays (e.g., small-world networks) scale logarithmically in terms of node degrees, propagation delay, and so on, but pub/sub routing on these structured overlays often exerts considerable overhead on each node for forwarding irrelevant messages. On the other hand, the unstructured *topic-connected overlay* (TCO) can eliminate unnecessary pure forwarders, since each topic induces a connected sub-overlay among all nodes interested in this topic; however, the node degrees and diameters are unbounded in a constructed TCO.

To achieve the best of both worlds, we design a practical pub/sub system in peer-to-peer settings, including both routing protocols and overlay topologies. First, based on small-world overlays, we propose the Nearest Subscribers and Matched Fingers (NSMF) routing protocol for topic-based pub/sub. Second, to reduce the routing overhead of NSMF, we construct small-world overlays that aim to maximize *interest closeness*, where each node strives to point its small-world fingers to nodes with common interests.

We validate our design with empirical evaluation and show the advantages, in both routing efficiency and overlay quality. As compared to regular small-world networks, our system reduces over 30% of the costs in both pure forwarding messages and the average path length.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols; C.2.4 [**Computer-Communication Networks**]: Distributed Systems; H.4.3 [**Information Systems Applications**]: Communications Applications

General Terms

Design, Experimentation, Performance

Middleware Industry'15 , December 07-11, 2015, Vancouver, BC, Canada

DOI: http://dx.doi.org/10.1145/2830013.2830017 .

Keywords

publish/subscribe, routing, overlay, small-world networks

1. INTRODUCTION

Publish/Subscribe (pub/sub) systems constitute an attractive choice as communication paradigm and messaging substrate for building large-scale distributed systems. This work concentrates on the *topic-based pub/sub* model: the system manages messages in categories named topics, publishers associate each publication message with one or more specific topics, and subscribers register their interests in a subset of all topics. Many real-world applications adopt topic-based pub/sub for message dissemination [1, 2, 12, 19].

A distributed pub/sub system often organizes nodes (e.g., brokers, servers or routers) in a federated or peer-to-peer manner as an overlay at the application or network layer. Once an overlay network is defined, the pub/sub system relies on some routing protocol to build message dissemination paths and to deliver publications to all subscriber nodes. Typically, a pub/sub routing protocol defines the forwarding function, which determines the set of next-hops for an incoming publication message at any node. Overlay topologies and routing protocols are closely related – both impact the performance and scalability of the pub/sub system.

It is of fundamental challenge to seamlessly tie both routing and overlay together in the system design. Many pub/sub implementations concentrate on one aspect only. For example, some pub/sub systems just employ the most generic overlays (e.g. a tree or full-mesh) and push all weights on the routing protocols. These pub/sub routing protocols are naturally difficult to scale by design, because they inevitably rely on sophisticated matching engines and large forwarding tables. Recently, many pub/sub systems try to migrate part of the complexity to overlay design. A well-constructed overlay could potentially simplify the pub/sub routing protocols and improve the efficiency of message dissemination.

Pub/sub overlay design poses unique challenges for distributed systems, such as the support of pub/sub routing and scalability with the number of topics (i.e., diverse interests), subscription size, and the volume of publications. We can generally classify existing pub/sub overlays into two categories: (1) structured and (2) unstructured. On the one hand, structured overlays, e.g., small-world networks [15, 17, 28], are scalable and robust with respects to node degrees, the path length of end-to-end delivery, and so on, but pub/sub routing over these structured overlays often exerts considerable dissemination overhead on each node for forwarding messages in which the node is uninterested [5, 13,

14, 24]. On the other hand, many unstructured overlays strive to minimize the amount of irrelevant messages being forwarded for pub/sub routing [4, 11, 23, 27]. In particular, *topic-connected overlay* (TCO) [6, 7, 8, 9, 10, 11, 22] can eliminate unnecessary pure forwarders, since each topic induces a connected sub-overlay among all nodes interested in this topic; however, it is challenging to bound node degrees or diameters for a TCO, even assuming centralized control and global knowledge. While resource constraints are obvious and mandatory in the context of peer-to-peer, mobile or sensor networks, the original motivation for TCO [10, 11] stem from a bounded-degree requirement in the cloud environments, where full-mesh overlays proved unscalable to connect application servers in an IBM production data center.

This work combines these two methods and attempts to gain merits from both. We incorporate basic principles of small-world networks and TCOs into a practical design for pub/sub systems in a peer-to-peer manner, including both routing protocols and overlay topologies. We show analytically and empirically that our pub/sub routing protocol achieves vast improvement on *small-world and interest-close overlays* as compared to regular small-world networks; in particular, our system reduces over 30% of the costs in both pure forwarding messages and the average path length.

2. RELATED WORK

In order to improve distributed pub/sub system performance and scalability, two directions have crystallized in the literature: (1) the design and implementation of routing protocols such that publications and subscriptions are distributed in a most efficient way across the overlay network (see [18, 29]) and (2) the construction of the overlay topology such that network traffic is minimized (e.g., [6, 10, 14, 22, 16]).

A significant body of research has been centered around either routing or overlay alone. Many earlier pub/sub systems put all weights on the routing protocols only, assuming the most naive overlays, such as trees or full-meshes. To support pub/sub routing, many nodes have to maintain a global view of all subscriptions on all nodes in the system. Consequently, such pub/sub systems suffer from large forwarding tables, excessively high matching complexity, need for selective message flooding, expensive routing computations, etc.

Recently, many pub/sub systems make efforts to migrate part of the complexity from routing protocols to overlay design. We can generally classify current pub/sub overlays into two major categories: (1) structured and (2) unstructured.

The structured overlays have been widely used in a variety of distributed systems. In particular, the structures of small-world networks (originated in [17]) inspired several popular DHT designs, e.g., Chord [28] and Symphony [20]. Small-world networks and DHTs also provide solid groundwork for our system design. Still, pub/sub overlays are fundamentally different from these canonical distributed overlay networks. For example, a DHT [15, 17, 20, 26, 28] maps IDs (as keys) to nodes – this mapping is self-dependent but determines the overlay topology and routing scheme; DHTs are ID-centric and thus inappropriate for organizing nodes that are semantically related, while pub/sub also needs to accommodate additional semantic information, e.g., topic interests at each node. As a result, pub/sub overlay design poses unique challenges for distributed systems, such as (a) construction of a semantic overlay for pub/sub; (b) support of routing protocols for subscription placement, interest matching, message dissemination, etc.; and (c) scalability with the number of topics (i.e., diverse interests), the subscription size, and the volume of publications.

Unstructured overlays are also appealing candidates for pub/sub overlay design. The TCO property is explicitly enforced in [4, 11, 23, 27] and implicitly manifest in [3, 14, 24, 25], because topic-connectedness effectively reduces unnecessary intermediate overlay hops for message delivery. However, it is unrealistic to keep node degrees or diameters of the TCO bounded, even if we can apply the state-of-the-art centralized algorithms [6, 7, 8, 9, 10, 22].

3. ROUTING ON SMALL-WORLD

3.1 Small-world networks for pub/sub

We abstract a distributed topic-based pub/sub system as an instance (V, T, I), where V is the set of nodes, T is the set of topics, and I is the interest function s.t. $I : V \times T \to \{0, 1\}$. Node $v \in V$ is interested in topic $t \in T$ iff $I(v, t) = 1$. We also say that node v subscribes to topic t.

Our pub/sub system design relies on a one-dimensional small-world network model [20, 26, 28]. We arrange all nodes along a ring and equip each node with a handful of small-world fingers, whose distributions are inversely proportional to the small-world distances.

Each node $v \in V$ has a unique identifier (i.e., nodeId) on a one-dimensional cyclic identifier space. For clarity and conciseness, we assume that every nodeId is $\log |V|$ bit, and consequently one node is present for every identifier in the space. Practical systems typically use $\log N$ bit nodeIds where $N \gg |V|$, so nodes do not fully populate the entire identifier space, which is essential for churn handling, e.g., node joins and departures. This assumption facilitates the presentations while not affecting the correctness of our system design [15], especially since this work does not consider dynamic churn in the system. Note that our pub/sub system design inherits churn resilience from small-world networks (see §3 and §4), which has been extensively studied [15, 20, 26, 28].

The small-world distance from node v to node w, which we denote by $\texttt{swDist}(v, w)$, is the clockwise numeric distance from v to w on the circle[1]. We say that node w (or edge $e = (v, w)$) is in the i-th small-world phase of node v, if $\texttt{swDist}(v, w) \in \left[2^i, 2^{(i+1)}\right)$, i.e., $w \in \left[v + 2^i, v + 2^{(i+1)}\right)$; we also denote the small-world phase as $v.\texttt{swPhase}(v, w) = v.\texttt{swPhase}(e) = i$.

PROPERTY 1. *In a network of the node set V, each node $v \in V$ maintains $k = \Theta(\log |V|)$ small-world fingers, and its i-th finger, $v.swFinger[i]$ where $0 \le i < k$, points to a node in the i-th small-world phase of v, i.e., $v.swFinger[i] \in \left[v + 2^i, v + 2^{(i+1)}\right)$.*

In the discussion that follows, we refer to small-world networks specifically as overlays with Property 1. Note that many DHTs attain this property [20, 28]. In Chord [28],

[1] We can also define $\texttt{swDist}(v, w)$ as the absolute distance between v and w, i.e., minimum of the clockwise and counter-clockwise distances. This difference only affects the constant factors hidden behind the Big-O notations [15, 20].

Data Structure 1 Publication messages

MESSAGE: a publication message transmitting between nodes

// publication information
- *topic*: the topic associated with the publication
- *content*: the content of the publication

// internal fields for pub/sub routing
- *low*: the lowest point of the targeted range, inclusive
- *high*: the highest point of the targeted range, exclusive

Data Structure 2 Routing table at node $v \in V$

▷ $v.swFinger$: the finger list indexed by the small-world phase in the id space, $v.swFinger[i] \in \left[v + 2^i, v + 2^{(i+1)}\right), 0 \leq i < \log|V|$.
▷ $v.nearestSub$: a hashtable that maps each topic $t \in T$ to $v.nearestSub(t)$, the subscriber of t that is nearest to v clockwise.

for instance, node $v \in V$ keeps exactly $k = \log|V|$ fingers, where $v.swFinger[i] = v + 2^i, 0 \leq i < k$. As [15] points out: although Chord defines specific small-world fingers for each node, this rigidity is not critical, and small-world networks allow flexibility for finger selection; specifically, routing between any two nodes is still $\mathcal{O}(\log|V|)$ hops, even if node $v \in V$ picks $v.swFinger[i]$ as any node in the range $\left[v + 2^i, v + 2^{(i+1)}\right), 0 \leq i < k$.

3.2 Pub/Sub routing on small-world networks: Nearest Subscribers and Matched Fingers

Before proposing our pub/sub routing protocol on small-world networks, we now introduce Data Structure 1 and 2.

Data Structure 1 defines the publication message for our pub/sub routing protocol. Given a message msg, $msg.topic$ and $msg.content$ together describe the data in a publication. Fields $msg.low$ and $msg.high$ jointly define the targeted range of msg, i.e., msg should reach all matched subscribers in the range $[msg.low, msg.high)$. For example, an initial publisher at $v \in V$ always specifies the target of the publication message to span the whole identifier space, i.e., $[v, v)$. Note that $msg.low$ also indicates the first destination of the message.

Data Structure 2 presents the local routing table at each node $v \in V$. First, node v keeps a list of small-world fingers, $v.swFinger$, which corresponds to Property 1; we will discuss how to construct these finger lists in §4. Second, node v maintains in $v.nearestSub$ the nearest subscriber for each topic $t \in T$; we assume that a membership service is available to provide such knowledge, which is beyond the scope of this work.

Protocol 3 specifies our routing protocol for pub/sub, Nearest Subscribers and Matched Fingers (NSMF). Upon receiving an incoming message msg at node v, Protocol 3 is responsible for delivering msg to all matched subscribers in the targeted range $[msg.low, msg.high)$. Line 2 checks whether v is $msg.low$, the first destination of msg: if not, Line 16 simply invokes `send`(msg), which forwards msg to one of its fingers that is nearest to $msg.low$; if $v = msg.low$, then Lines 3-14 spread this message to an array of next-hops, X, that collectively cover the targeted range. Line 3 sets $t = msg.topic$, and Line 4 computes X by combining (1) all fingers that subscribe to t and lie in the targeted range of msg and (2) the nearest subscriber for t. Line 5 first sorts all next-hops in X in ascending order according to the small-world distances from v, and thus $X_0 = v.nearestSub(t)$.

Protocol 3 Nearest Subscribers and Matched Fingers (NSMF) routing for pub/sub at node $v \in V$

v.NSMF()
1: **upon** receiving a message msg
2: **if** $v = msg.low$ **then**
3: $t \leftarrow msg.topic$
4: $X \leftarrow \{u \in v.swFinger | I(u,t) \wedge u \in [msg.low, msg.high)\} \bigcup \{v.nearestSub(t)\}$ // initialize X, the next-hop array
5: sort X in ascending order with regard to small-world distances from v
6: **if** $|X| > 0$ **then**
7: **for all** $j = 0, \ldots, |X| - 1$ **do**
8: $msg_j \leftarrow$ clone msg
9: $msg_j.low \leftarrow X_j$
10: **if** $j + 1 < |X|$ **then**
11: $msg_j.high \leftarrow X_{j+1}$
12: **else**
13: $msg_j.high \leftarrow msg.high$
14: `send`(msg_j)
15: **else**
16: `send`(msg)

v.`send`(msg)
1: forward msg to v's finger that is nearest to $msg.low$

Then, Lines 6-14 prepare a message msg_j for every next-hop X_j, $j = 0, \ldots, |X| - 1$. Each X_j is in charge of a sub-range as specified in msg_j, and the union of all these sub-ranges spans the entire targeted range of msg $[msg.low, msg.high)$: X_0, ..., X_{d-2}, and X_{d-1} cover $[X_0, X_1)$, ..., $[X_{d-2}, X_{d-1})$, and $[X_{d-1}, msg.high)$, respectively, where $d = |X|$. We skip $[msg.low, X_0) = [v, v.nearestSub(t))$, because this sub-range contains no subscriber of t. Following Line 14, X_j will receive msg_j from v and recursively invoke Protocol 3. These recursive processes continue, until the publication message reaches all subscribers in the targeted range.

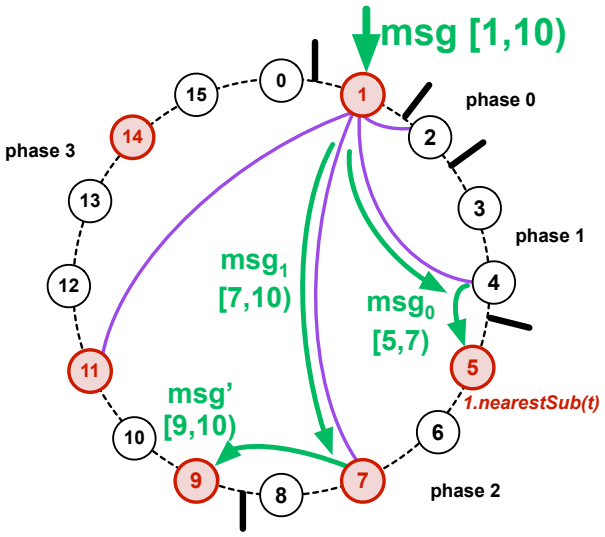


Figure 1: A running example for Protocol 3

In Fig. 1, we have a small-world network of 16 nodes, i.e., $V = \{0, 1, \ldots, 15\}$. The figure highlights the small-world fingers of node 1, which connects to node 2, 4, 7, and 11,

respectively at each phase. For some topic $t \in T$, nodes that subscribe to t are $V^{(t)} = \{1, 5, 7, 9, 11, 14\}$, which Fig. 1 illustrates as shaded vertices. Suppose that node 1 receives a message *msg* on t with the targeted range $[1, 10)$, which we simply denote as $msg\,[1, 10)$. Node 1 should send *msg* to all nodes that are interested in t and in the range of $[1, 10)$, i.e., $\{5, 7, 9\}$. After Line 5, node 1 obtains the sorted next-hop array $X = \{5, 7\}$: $X_0 = 5$ is the nearest subscriber for t, and $X_1 = 7$ is a matched finger; X excludes the other matched finger, node 11, because $11 \notin [1, 10)$. Node 1 prepares $msg_0\,[5, 7)$ for $X_0 = 5$ and $msg_1\,[7, 10)$ for $X_1 = 7$. Since node 1 has no finger linking to $X_0 = 5$, it sends msg_0 to node 4, which then forwards msg_0 to node 5. Node 1 directly sends msg_1 to node $X_1 = 7$. Upon receiving msg_0 and msg_1, both node 5 and 7 call Protocol 3, respectively. Node 5 sends no further messages, since its nearest subscriber (node 7) is out of the targeted range of $msg_0\,[5, 7)$. Node 7 sends a single message $msg'\,[9, 10)$ to its nearest subscriber 9. The recursive processes end at node 9, since node 9 is the only one in the targeted range of $msg'\,[9, 10)$. In summary, NSMF accomplishes pub/sub routing by disseminating the publication message to all subscribers in the targeted range.

LEMMA 1. *Protocol 3 guarantees to eventually deliver each publication message to all subscribers in the targeted range, if the overlay is a small-world network with Property 1.*

4. SMALL-WORLD AND INTEREST-CLOSE OVERLAY

We focus on how to construct the overlays that best suit our pub/sub routing protocol NSMF. First, to ensure correctness of Protocol 3, the overlay should be a small-world network with Property 1 (see Lemma 1). Second, we hope the overlay to yield minimum *routing overhead*, which this work and many others [4, 11, 13, 14, 23, 24, 27] define to be the total number of pure forwarding messages, i.e., the messages that each node $v \in V$ receives yet has no interest in.

Intuitively, diminishing the routing overhead prefers *interest closeness*, meaning that the overlay places nodes with similar interests close to each other. Topic-connected overlay (TCO) is one way to realize interest closeness, which promises to be capable of eliminating the routing overhead in pub/sub [6, 10, 11]. Informally speaking, TCO organizes all nodes interested in the same topic in a directly connected dissemination sub-overlay. TCO can support the transmission of publications on each topic to all subscribers without using non-interested nodes as intermediate relays. Pub/sub routing atop TCOs saves bandwidth and computational resources otherwise wasted on forwarding and filtering out unwanted messages. Unfortunately, TCO and Property 1 are at odds with each other. More specifically, it is NP-hard whether we can construct a TCO with a fixed average or maximum node degree [6, 7, 10, 22], while Property 1 strictly restricts each node to possess a bounded number of small-world fingers. Therefore, we decide to relax the TCO requirement by approximating a TCO.

Formally speaking, given an instance (V, T, I), we regard the pub/sub overlay as an undirected graph $G = (V, E)$ over the node set V with the edge set $E \subseteq K$, where we denote by K the ground set of all possible edges among V, i.e., $K = V \times V$. Given $G = (V, E)$, the sub-overlay *induced* by $t \in T$ is a subgraph $G^{(t)} = (V^{(t)}, E^{(t)})$ such that $V^{(t)} = \{v \in V | I(v, t) = 1\}$ and $E^{(t)} = \{(v, w) \in E | v \in V^{(t)} \wedge w \in V^{(t)}\}$. A topic-connected component (TCC) on topic $t \in T$, is a maximal connected subgraph in $G^{(t)}$. If $G^{(t)}$ contains at most one TCC for each topic $t \in T$, then $G = (V, E)$ forms a *topic-connected overlay* (TCO) for (V, T, Int), which we denote as $TCO(V, T, Int, E)$.

Given an edge set $E \subseteq K$, $TCC(E)$ stands for the total number of TCCs in $G = (V, E)$ over all topics T,

$$TCC(E) = \sum_{t \in T} \left(\#\text{TCCs in } G^{(t)} = (V^{(t)}, E^{(t)}) \right) \quad (1)$$

By definition,

$$TCC(\emptyset) = \sum_{t \in T} \left| V^{(t)} \right| \quad (2)$$

$$TCC(K) = \left| \{t \in T : V^{(t)} \neq \emptyset\} \right| \quad (3)$$

$$TCC(E) = TCC(K) \text{ iff } E \subseteq K \text{ forms a TCO} \quad (4)$$

We can use $TCC(E)$ to measure the progress towards TCO: suppose E is initially empty and grows by adding edges one by one, then $TCC(E)$ starts from $TCC(\emptyset)$ and strictly decreases with every edge addition down to an absolute limit, i.e., $TCC(K)$.

Further, we define the *contribution* (towards TCO) of edge e with respect to a given edge set E to be the number of TCCs that would be reduced by adding e upon E.

$$contrib_E(e) = TCC(E) - TCC(E + e) \quad (5)$$

Then, we have Lemma 2:

LEMMA 2. *Given an instance (V, I, T) and an edge set $E \subseteq K$, we place an arbitrary order on all edges in E, i.e., $E = \{e_1, e_2, \ldots, e_m\}$. Let $E_0 = \emptyset$ and $E_i = \{e_1, ..., e_i\}, 1 \leq i \leq m$, then E forms a TCO iff*

$$\sum_{1 \leq i \leq m} contrib_{E_{i-1}}(e_i) = TCC(\emptyset) - TCC(K)$$

As Lemma 2 shows, $TCC(\emptyset) - TCC(K)$ represents the amount of TCCs that $E \subseteq K$ should reduce to achieve a TCO. We define the *TCO support ratio* for an overlay edge set E:

$$TcoSuppR(E) = \frac{TCC(\emptyset) - TCC(E)}{TCC(\emptyset) - TCC(K)}, \forall E \subseteq K \quad (6)$$

The TCO support ratio $TcoSuppR$ can serve as an indicator of the overlay quality: (1) $TcoSuppR(E) \in [0, 1]$, $TcoSuppR(\emptyset) = 0$, and $TcoSuppR(E) = 1$ iff E forms a TCO; and (2) $TcoSuppR(E)$ is non-decreasing as E expands, i.e., $TcoSuppR(E) \leq TcoSuppR(E')$, if $E \subseteq E'$ Thus, the higher $TcoSuppR(E)$ is, the closer E approximates a TCO.

We formalize the problem of small-world and interest-close overlay design as follows:

PROBLEM 1. *Given an input instance (V, T, I), construct an edge set $E \subseteq K$ that maximizes $TcoSuppR$ while forming a small-world network with Property 1.*

Alg. 4 specifies GrSwico, a greedy algorithm for Problem 1. Alg. 4 starts from an empty edge set $E = \emptyset$ and constructs E iteratively, until attaining a small-world network with Property 1. At each iteration, Alg. 4 greedily adds to E from

Algorithm 4 Greedy heuristic for Small-World and Interest-Close Overlay (GrSwico)

GrSwico(V, T, I)

Input: (V, T, I)
Output: E, which forms a small-world network for V

```
1: E ← ∅, P ← V × V
2: while P ≠ ∅ do
3:    e' ← arg max_{e∈P} contrib(e)
4:    E ← E + {e'}
5:    P ← P − {e'}
6:    for all e : v.swPhase(e) = v.swPhase(e') where v ∈ e do
7:       P ← P − {e}
8: return E
```

the potential edge set P an edge e' with the highest contribution (see Lines 3-4). After adding e', Property 1 enforces Lines 5-7, which remove from P all edges that belong to the same small-world phase as e', because there is only one spot for each small-world phase. Suppose, for example, Alg. 4 adds edge $(1, 7)$ in Fig. 1 at some iteration, then we need to remove from P some edges, such as $\{(1, 5), (1, 6), (1, 8)\}$, which are also in phase 2 of node 1. Based on [6, 8], we prove that Alg. 4 is correct and runtime efficient:

LEMMA 3. *Alg. 4 outputs an edge set E for Problem 1 in time $\mathcal{O}(|V|^2|T|)$.*

Lemma 3 shows that GrSwico produces a small-world network with Property 1. Therefore, our designed overlays possess all nice properties of small-world networks, e.g., logarithmic latency for end-to-end message delivery, and churn resilience [15, 20, 26, 28].

Alg. 4 is a static and centralized algorithm for pub/sub overlay design. However, GrSwico provides a baseline for more sophisticated approaches (e.g., decentralized protocols), because it dominates typical small-world networks for pub/sub routing (see §5).

5. EVALUATION

5.1 Experiment setup

We implement all algorithms and routing protocols in PeerSim [21]. We also develop two small-world overlays as comparison baselines for GrSwico:

(1) Chord [28], a DHT that always selects the finger as the first node in each small-world phase, i.e., $v.swFinger[i] = (v + 2^i)$, $0 \le i < \log|V|$, $\forall v \in V$.

(2) RandomSW, a small-world network that randomly chooses the finger in each small-world phase: all nodes in the range $\left[v + 2^i, v + 2^{i+1}\right)$ are equally probable to become $v.swFinger[i]$, $0 \le i < \log|V|$, $\forall v \in V$.

The overlays produced by GrSwico, Chord and RandomSW have the same node degrees and preserve Property 1. As a result, the expected path length to route between any two nodes is $\mathcal{O}(\log|V|)$ hops in all three overlays [15].

We synthetically generate the input instances (V, T, I) as follows: $|V| \in [1\,000, 10\,000]$, $|T| \in [1\,000, 10\,000]$, and each node $v \in V$ has a fixed subscription size $|\{t \in T | I(v, t) = 1\}| = 20$. We associate each topic $t \in T$ with $p(t)$ such that $\sum_t p(t) = 1$, and each node subscribes to t with probability $p(t)$. The value of $p(t)$ follows either a uniform, a Zipf (with $\alpha = 2.0$), or an exponential distribution, which we call Unif, Zipf, or Expo, respectively. These distributions are representative of actual workloads used in industrial pub/sub systems today [10, 11]. Stock market monitoring engines use Expo for the study of stock popularity in the New York Stock Exchange [30]. Zipf faithfully describes the feed popularity distribution in RSS feeds [19].

Our publication workloads are uniformly random with respect to the topics and all subscribers. We issue publication messages iteratively. At each iteration, the system publishes one message on some topic $t \in T$ from an issuer node $v \in V$: every topic $t \in T$ possesses the same probability $1/|T|$, and all subscribers of t have the equal chance to be the issuer.

We evaluate three basic metrics: (a) the routing overhead (see definition in §4), (b) *TcoSuppR* in Eq. (6), and (c) the average path length across all messages, where the path length of a message *msg* is the number of hops for *msg* to travel from its source to the destination *msg.low*.

5.2 Impact of $|V|$

Fig. 2 depicts the comparison among GrSwico, Chord, and RandomSW under Unif as $|V|$ increases from 1 000 to 10 000, where we fix $|T| = 2\,000$.

We look at the routing overhead. Fig. 2(a) plots the comparable ratios of the routing overheads for Protocol 3 on various small-world overlays. GrSwico consistently yields substantially less routing overhead: Chord and RandomSW exert additionally 32% and 29% more pure forwarding messages, respectively on average.

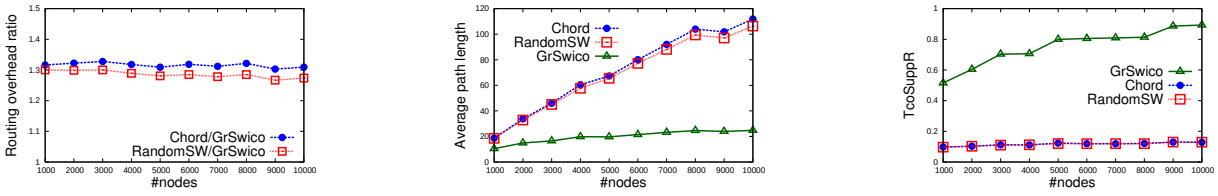
We look at the average path length. Fig. 2(b) shows that GrSwico exhibits remarkably better scalability than either Chord or RandomSW. First, the average path lengths of both Chord and RandomSW are more than 3.3 times that of GrSwico, on average. Second, the margins between GrSwico and other two overlays become even wider when the workloads scale up. The average path lengths of all are enlarging with the number of nodes. However, GrSwico grows at a steadily slow pace, while Chord and RandomSW increase almost linearly as the node set V expands. At $|V| = 10\,000$, the average path lengths of GrSwico, Chord, and RandomSW are 24.7, 112, and 107, respectively.

We look at the TCO support ratio. In Fig. 2(c), GrSwico achieves significantly higher *TcoSuppR*: GrSwico is 75.3%, and Chord and RandomSW are around 11.6%, on average. Besides, this benefit becomes more profound as the node set grows: the *TcoSuppR* of GrSwico increases with the number of nodes, while those of Chord and RandomSW stay consistently low. The root cause lies in the subscription correlation among all nodes: increasing the number of nodes enriches the correlation. GrSwico appreciates this upgraded correlation, because each node has a better chance to find small-world fingers with larger interest overlaps; meanwhile, Chord and RandomSW disregard the correlation and thus gain none improvement.

5.3 Impact of $|T|$

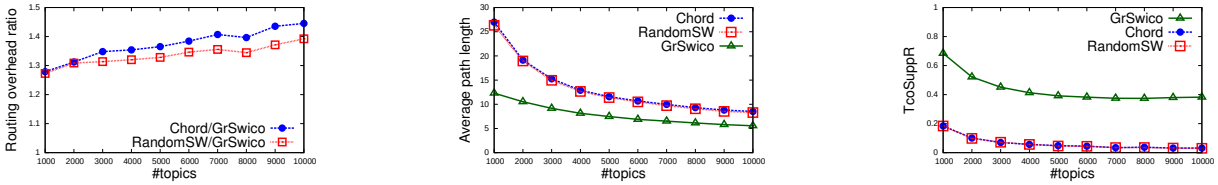
Fig. 3 shows how GrSwico, Chord, and RandomSW perform with $|T|$ under Unif. In this setting, $|V| = 1\,023$, and $|T|$ varies from 1 000 to 10 000.

Fig. 3(a) plots relative ratios of the routing overheads on Chord and RandomSW as compared to that on GrSwico. GrSwico significantly outperforms both baselines: Chord and RandomSW impose 37.1% and 33.5% more pure forwarding messages than GrSwico, respectively on average. This



(a) Routing overhead (b) Average path length (c) TCO support ratio

Figure 2: GrSwico versus Chord versus RandomSW wrt. $|V|$



(a) Routing overhead (b) Average path length (c) TCO support ratio

Figure 3: GrSwico versus Chord versus RandomSW wrt. $|T|$

performance advantage is increasing as the input instance scales up with the number of topics. As $|T| = 10\,000$, the additional amount of pure forwarding messages is 44.5% for Chord and 39.3% for RandomSW.

Fig. 3(b) depicts the average path lengths for different small-world overlays. GrSwico takes only about 60% of the average path lengths of others: GrSwico is 7.8 hops, while Chord and RandomSW are over 13 hops, on average. It is worth mentioning that the gaps between GrSwico and the other two shrink as the number of topics increases. GrSwico gains benefits over others by exploiting the correlation embedded in the subscriptions of all nodes, and this correlation is diminishing as a result of enlarging the topic set. We believe that, as $|T| \to \infty$, these three curves will converge to the same value, $\log |V|$. In this extreme case, nodes share virtually none common interest, and all small-world networks are equivalent in the sense that zero interest closeness is captured; each pub/sub message dissemination involves exactly one small-world routing from the source to a singleton destination, so the expected path length is around $\log |V|$ hops.

We further explain the marginal advantages of GrSwico against ordinary small-world networks by looking at the TCO support ratios in different overlays. As Fig. 3(c) shows, *TcoSuppR* of GrSwico is considerably better than others: GrSwico achieves up to 44%, and both Chord and RandomSW are merely around 6%, on average. Higher TCO support indicates better chances to have matched fingers for disseminating publication messages. As a result, the messages rely on fewer intermediate pure relays to move from the original issuer to the subscribers.

6. CONCLUSIONS

This work demonstrates the importance and benefits of integrating routing protocols with overlay topologies for distributed pub/sub systems. Furthermore, it is still of great potential to improve the qualities of the small-world and interest-close overlays, which provides us a solid guidance and a promising direction for future work.

7. ACKNOWLEDGMENT

This research has been partially supported by European Commission (EU) under the FP7 Programme Projects: iSocial (No. 316808) and COMPOSE (No. 317862).

8. REFERENCES

[1] Apache Kafka. `http://kafka.apache.org/`.

[2] IBM IoT Foundation. `http://internetofthings.ibmcloud.com/`.

[3] E. Baehni, P. Eugster, and R. Guerraoui. Data-aware multicast. In *DSN'04*.

[4] R. Baldoni, R. Beraldi, V. Quema, L. Querzoni, and S. Tucci-Piergiovanni. TERA: topic-based event routing for peer-to-peer architectures. In *DEBS*, 2007.

[5] M. Castro, P. Druschel, A.-M. Kermarrec, and A. Rowstron. SCRIBE: A large-scale and decentralized application-level multicast infrastructure. *JSAC*, 2002.

[6] C. Chen, Y. Tock, H.-A. Jacobsen, and R. Vitenberg. Weighted overlay design for topic-based publish/subscribe on geo-distributed data centers. In *ICDCS*, 2015.

[7] C. Chen, R. Vitenberg, and H.-A. Jacobsen. Brief announcement: Constructing fault-tolerant overlay networks for topic-based publish/subscribe. In *PODC' 13*.

[8] C. Chen, R. Vitenberg, and H.-A. Jacobsen. A generalized algorithm for publish/subscribe overlay design and its fast implementation. In *DISC'12*.

[9] C. Chen, R. Vitenberg, and H.-A. Jacobsen. Scaling construction of low fan-out overlays for topic-based publish/subscribe systems. In *ICDCS'11*.

[10] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg. Constructing scalable overlays for pub-sub with many topics: Problems, algorithms, and evaluation. In *PODC*, 2007.

[11] G. Chockler, R. Melamed, Y. Tock, and R. Vitenberg. Spidercast: a scalable interest-aware overlay for topic-based pub/sub communication. In *DEBS*, 2007.

[12] B. F. Cooper, R. Ramakrishnan, U. Srivastava,

A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. Pnuts: Yahoo!'s hosted data serving platform. *Proc. VLDB Endow.*, 2008.

[13] S. Girdzijauskas, G. Chockler, R. Melamed, and Y. Tock. Gravity: An interest-aware publish/subscribe system based on structured overlays. In *DEBS*, 2008.

[14] S. Girdzijauskas, G. Chockler, Y. Vigfusson, Y. Tock, and R. Melamed. Magnet: practical subscription clustering for internet-scale publish/subscribe. In *DEBS*, 2010.

[15] K. Gummadi, R. Gummadi, S. Gribble, S. Ratnasamy, S. Shenker, and I. Stoica. The impact of dht routing geometry on resilience and proximity. In *Sigcomm'03*.

[16] M. A. Jaeger, H. Parzyjegla, G. Mühl, and K. Herrmann. Self-organizing broker topologies for publish/subscribe systems. In *SAC'07*.

[17] J. Kleinberg. The small-world phenomenon: An algorithmic perspective. In *STOC*, 2000.

[18] G. Li, V. Muthusamy, and H.-A. Jacobsen. Adaptive content-based routing in general overlay topologies. In *Middleware'08*.

[19] H. Liu, V. Ramasubramanian, and E. G. Sirer. Client behavior and feed characteristics of RSS, a publish-subscribe system for web micronews. In *IMC*, 2005.

[20] G. S. Manku, M. Bawa, and P. Raghavan. Symphony: Distributed hashing in a small world. In *USITS*, 2003.

[21] A. Montresor and M. Jelasity. PeerSim: A scalable P2P simulator. In *P2P*, 2009.

[22] M. Onus and A. W. Richa. Parameterized maximum and average degree approximation in topic-based publish-subscribe overlay network design. In *Icdcs'10*.

[23] J. A. Patel, E. Rivière, I. Gupta, and A.-M. Kermarrec. Rappel: Exploiting interest and network locality to improve fairness in publish-subscribe systems. *Computer Networks*, 2009.

[24] F. Rahimian, S. Girdzijauskas, A. H. Payberah, and S. Haridi. Vitis: A gossip-based hybrid overlay for internet-scale publish/subscribe enabling rendezvous routing in unstructured overlay networks. In *IPDPS*, 2011.

[25] F. Rahimian, T. Le Nguyen Huu, and S. Girdzijauskas. Locality-awareness in a peer-to-peer publish/subscribe network. In *DAIS*, 2012.

[26] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware'01*.

[27] V. Setty, M. van Steen, R. Vitenberg, and S. Voulgaris. Poldercast: Fast, robus, and scalable architecture for p2p topic-based pub/sub. In *Middleware'12*.

[28] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. In *Sigcomm'01*.

[29] M. A. Tariq, B. Koldehofe, and K. Rothermel. Efficient content-based routing with network topology inference. In *DEBS '13*.

[30] Y. Tock, N. Naaman, A. Harpaz, and G. Gershinsky. Hierarchical clustering of message flows in a multicast data dissemination system. In *IASTED PDCS*, 2005.